UNIVERSITY OF BUEA

P.O. Box 63,

Buea, South West Region

CAMEROON

Tel : (237) 3332 21 34/3332 26 90

Fax: (237) 3332 22 72

REPUBLIC OF CAMEROON

PEACE-WORK-FATHERLAND

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

COURSE: CEF 440 - Internet Programming and Mobile Programming

MOBILE APPLICATION DEVELOPMENT PROCESS

Presented by:

NAMES	MATRICULES
KENNE DATEWO SUZY MAIVA	FE21A214
SIAHA TOUKO AUBIN	FE21A304
DJEUNOU DJEUNOU MARIEKE JETTIE	FE21A168
TSAPZE ZAMBOU ROSELINE CYNTHIA	FE21A328
MATHO SONKWA HESTIE MAYELLE	FE21A438

Academic year: 2023-2024

COURSE INSTRUCTOR: Dr. NKEMENI VALERY

Contents

Table of figures

Table of table

List of abbreviation

APIs: Application of Programming Interfaces

Cpp: C plus plus

CSS: Cascading style sheet

HTML: Hypertext Markup Language

JS: JavaScript

KPIs: Key Performance Indicator

TS: TypeScript

UI: User interface

UX: user experience

INTRODUCTION

An application is a software that lets you exchange information with customers and help them complete specific tasks. Every day, the numbers of applications released to the market grows and with such promising, it is easy to see why many companies are looking into mobile apps development! Thus, focusing on consumer-facing projects and the revenue they can generate. Others want to optimize internal processes and choose to invest in enterprise solution that can reduce expenses and improve efficiencies. Whatever the case, if developing a mobile solution is of interest to our firm, it's valuable for us to know the process of building an app' and this can be drove through 7 key steps which are the strategy of development, the analysis and planning, UI/UX design, the app development (frontend, backend, middleware), after looking at the application testing, the deployment of the app and finally the support and performance Monitoring(KPIs).

I. MOBILE APP DEVELOPMENT PROCESS

1. Review of major types of mobile apps

Different types of applications are based on their development method and internal functionality. There are numerous categories of mobile apps on the market: games, social, travel, productivity, utilities, shopping, lifestyle, educational, health and fitness, and many more. They have become such a fixture of our daily lives that we rarely think about the mechanics and effort that goes into their creation. apps can be divided into three types: native, hybrid, and web apps.

Web apps are delivered over an internet browser. Users don't need to install them on their devices some example of web apps are Google Chrome, safari, Google photo, Canva. With this technology, any device with a web browser can be used to access a web app. This makes development and maintenance easier since you can build a single web app to work across multiple platforms; also, Web apps don't need to be downloaded from an app store, so they won't take up storage on a user's device. However, they do require internet access to work which can lead to slow load times and poor usability.

Native apps on the other hand, are built for a specific platform or device type or operating systems. In this type of application, the user must install the appropriate software version on their device of choice from an app store or a marketplace. For example, google play store for android or app store for iOS. Android apps are written in **Java or Kotlin** while Apple apps use **Swift**. This approach means native apps can interact with other device features, such as the microphone, camera, or push notifications. Now that there are more cross-platform tools like **Flutter and React Native**, native app development is more accessible for developers of various skill levels. Native mobile apps let users interact with their devices' internal hardware and operating systems. You can grant users access to native features like device location tracking, device microphone and cameras, user contact lists, touch gestures, device tilt, and other user interactions, device security features like a fingerprint scan or face recognition. Examples are, snapchat, Google play, apple music, google calendar, iMovie, etc.

Hybrid apps are native applications with a web browser embedded inside them (e.g.: Facebook, Instagram, Gmail, Discord). The main advantage of choosing hybrid apps is that development is streamlined since businesses only need to build one version of the product. This also means that the development process is typically quicker than that of native apps, as well as less costly. Apps can also function offline, and updates are easier to facilitate. These apps are written in cross-platform languages like HTML5, CSS, and JavaScript.

The table 1 bellow, from amazon give us a brief key differences and comparison of different types of applications:

Table 1: Summary of differences: web apps vs. hybrid apps vs. native apps

Characteristics	**Web app**	**Hybrid app**	**Native app**
Usage	Users can access directly from a browser	User have to install the app on their device of choice	Users have to install the app on their device of choice
Internal working	Client code in the browser communicates with remote server – side code and databases	Client code and browser code wrapped in natives shell or container	Client code writing in technology and language specific to the device or platform it will be installed on
Native device feature	Not accessible	Accessible	Accessible
User experience	Inconsistent and dependent on the browser being used	Consistent and engaging	Consistent and engaging
Access	Limited by browsers and network connectivity	One – step access with offline features	One – step access with offline features
Performance	Slower and less responsive	Faster, but may consume more battery power	Performance can be optimized to device
Development	Cost-efficient, faster time to market	Cost-efficient, faster time to market	Expensive, slower time to market

2. Mobile app programming languages and development frameworks

We all know, mobile app programming languages are the foundation of building applications for mobile devices. Native apps are developed specifically for a particular platform, such as iOS or Android, using languages like Swift or Objective-C for iOS and Java or Kotlin for Android. These languages provide direct access to the device's features and deliver high performance and seamless user experiences. Web apps, on the other hand, are built using web technologies like HTML, CSS, and JavaScript. They run within the device's web browser, offering cross-platform compatibility but with limited access to device features. Hybrid apps combine elements of both native and web apps, leveraging frameworks like React Native or Flutter. These frameworks allow developers to use web technologies to build the app's user interface while accessing device features through native APIs.

The mobile app landscape is booming, with new applications constantly emerging to meet our ever-evolving needs. But what goes into building these ubiquitous tools? This is where mobile app development frameworks come in. These frameworks act as the foundation upon which your app is built, providing a structure and set of tools to streamline the development process. Choosing the right framework is crucial for the success of your app. The following table 2, given by mobidev gives a brief recap of framework used in mobile development nowadays according to some key features:

Table 2: Comparison between Mobile app development frameworks

Feature	**React Native**	**Flutter**	**Xamarin**	**NativeScript**
Language	JS	Dart	C#	JS or TS
Performance	Good performance, can have some issues with complex animations.	High performance due to its compiled code, which can be optimized for each platform.	Good performance, as it compiles to native code.	Generally good performance, as it uses native components.
Cost &Time to Market	Relatively low cost and fast time to market due to code	Similar to React Native, with potentially	High cost due to licensing fees for some	Similar to React Native and Flutter, with

	reusability across platforms.	faster development due to its hot reload feature each platform.	features, but development time can be faster for teams familiar with C# and .NET.	reusable code across platforms.
UI & UX	Provides a native-like experience, with access to native components and modules.	Provides customizable UI components and offers a smooth, native-like experience.	Provides access to native APIs and UI components, offering a native user experience.	Allows access to native APIs and UI components, resulting in a native look and feel.
Complexity	Moderate complexity, especially for developers familiar with React	Moderate to high complexity, especially for developers new to Dart.	Moderate to high complexity, especially for developers new to C# and .NET.	Moderate complexity, especially for developers familiar with JavaScript.
Community Support	Large and active community with extensive documentation and third-party libraries.	Growing community with good documentation and increasing third-party package support.	Active community with good documentation and third-party library support.	Active community with decent documentation and plugin support.
Use Cases	Great for complex apps where a native feel is important, but development	Ideal for creating visually appealing apps	Perfect for enterprise-level apps that require deep	Suitable for cross-platform mobile app development

	speed is also a priority. Suitable for building cross-platform apps for iOS and Android such as Complex apps, social media, games (with limitations), e-commerce	with a rich user experience across platforms. Well-suited for MVP development, Complex apps, e-commerce	integration with native features and functionalities	for iOS and Android

3. Mobile application architectures and design patterns

Design patterns are reusable solutions to common problems in software design. Mobile application architectures and design patterns play a crucial role in organizing and structuring the codebase, improving maintainability, and enhancing the overall performance of mobile apps. Here are some common mobile application architectures and design patterns:

- Model-View-Controller (MVC):

MVC is a software architectural pattern that separates the application into three interconnected components: the Model (data and business logic), the View (UI), and the Controller (handles user interactions and updates the Model and View). The block diagram of MVC can be found in figure 1 bellow:

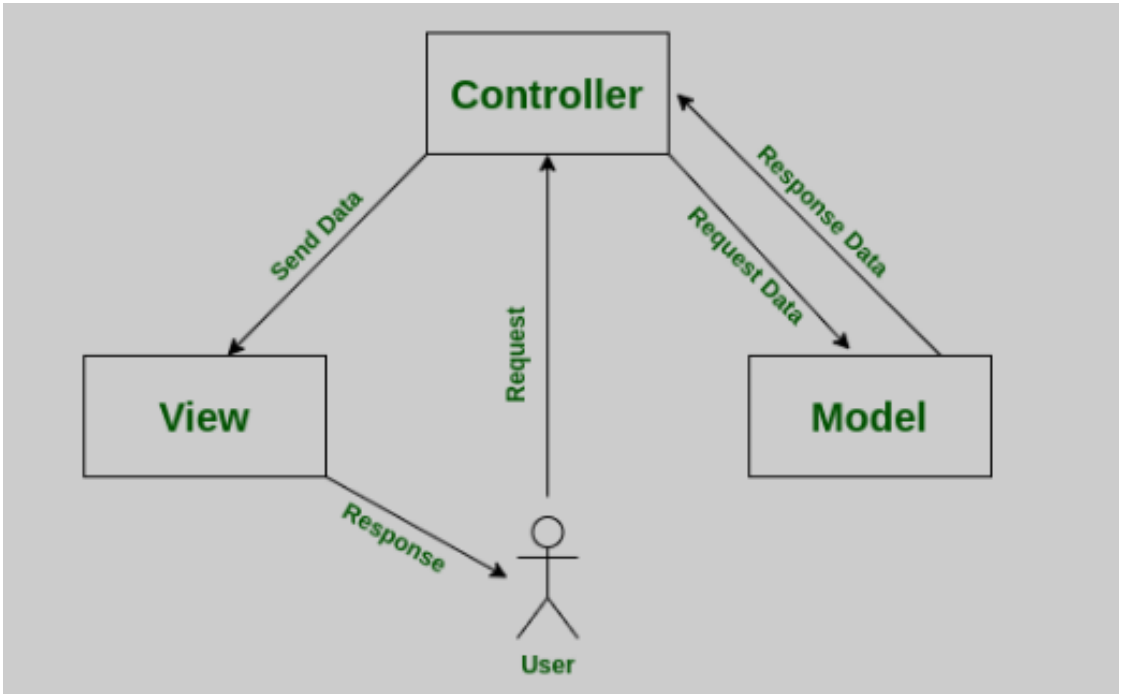

Figure 1: Model-View_Controller architecturechr

- Model-View-ViewModel (MVVM):

MVVM is an architectural pattern that focuses on the separation of concerns and data binding. The Model represents the data, the View handles UI rendering, and the View Model acts as an intermediary between the Model and the View, providing data and handling user interactions as shown in figure 2 bellow.

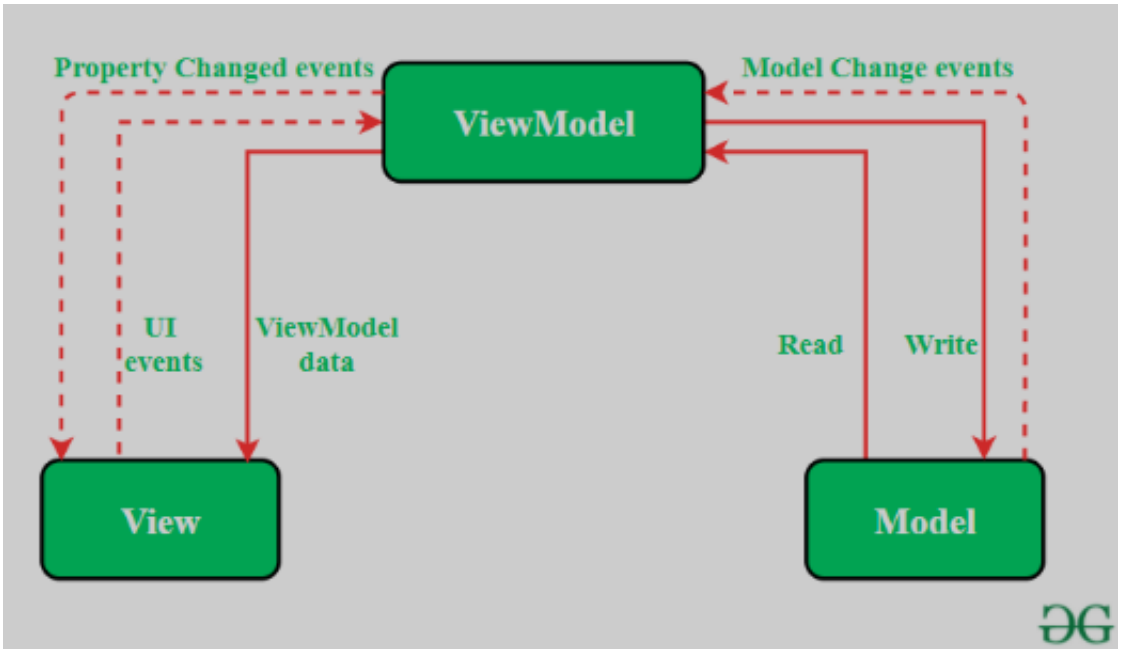

Figure 2: Model-View-ModelView architecture

- Clean Architecture:

Clean Architecture, as shown in figure 3 is an architectural approach that promotes separation of concerns and independence of implementation details. It divides the application into layers, such as Domain, Presentation, and Data, with clear boundaries and dependencies flowing inward. This architecture allows for easier testing, maintainability, and scalability.

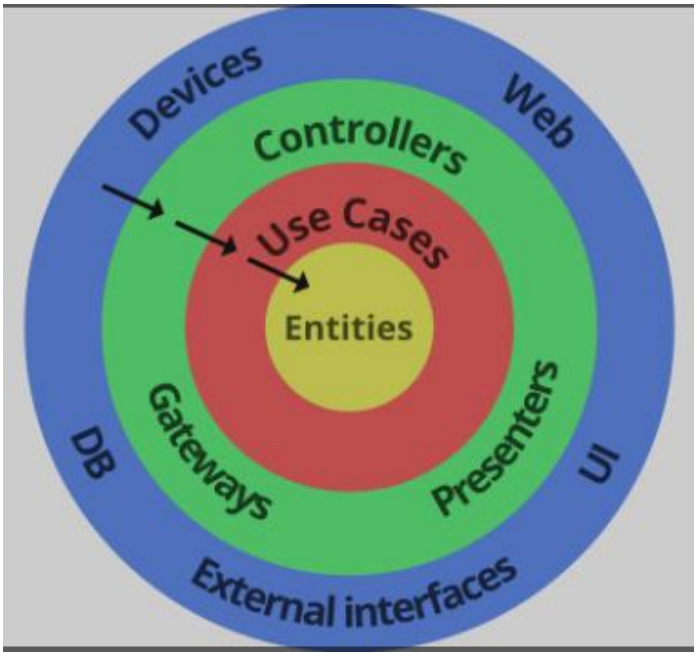

Figure 3: Clean architecture

- Single-Page Application (SPA):

SPA is an architecture that delivers a seamless user experience by loading the entire application in a single web page. It utilizes JavaScript frameworks like React or Angular to dynamically update the content without requiring page reloads.

- Microservices Architecture:

Microservices Architecture is an approach where an application is divided into small, independent services that can be developed, deployed, and scaled individually. Each service focuses on a specific business capability and communicates with other services through well-defined APIs.

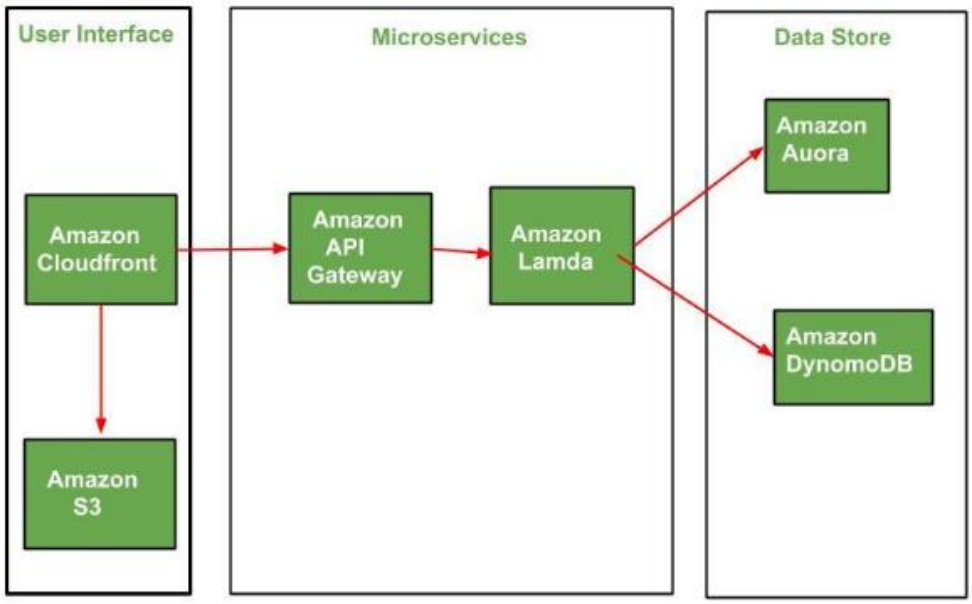

Figure 4: Microservices architecture

4. Collect and analyze user requirements

You can use various methods, such as surveys, interviews, focus groups, personas, user stories, and user journeys, to collect and analyze data about the users. User research helps you to design an app that solves the users' problems, meets their expectations, and provides a positive user experience. Good app requirements are the foundation of any app project and can sometimes be the difference between success and failure. Basically, there are 7 key steps for gathering good app requirements that can be listed as follow:

- **Identify the key stakeholders**: start by identifying the key stakeholders who will be impacted by the app, including business owners, users, and other decision-makers. Make sure to involve all relevant stakeholders in the requirements gathering process. Giving people a voice in the earliest stages of the project will also help with app adoption at the end of the project.
- **Understand the business needs:** understand the business needs that the app is intended to address. This can include understanding the goals and objectives of the business, as well as any challenges or pain points that the app should address. This will provide understanding of why the app and its features are important, not just what the app needs to do.
- **Identify the user requirements**: identify the requirements of the end-users who will be using the app. This can include understanding their needs, goals, and pain points, as well as any usability or

accessibility requirements. Despite all other planning efforts, there's no substitute for speaking with actual end-users.

- **Define the functional requirements:** define the functional requirements of the app, including the features and capabilities that are required. This can include requirements for data input and output, integration with other systems, and reporting and analytics.
- **Define the technical requirements:** define the technical requirements of the app, including the platforms, technologies, and infrastructure that will be used. This can include requirements for scalability, performance, and security. While gathering these requirements shouldn't be skipped, using a low-code cloud platform for development can greatly simplify the implementation, as the right platform will check all the boxes.
- **Prioritize the requirements:** Prioritize the app requirements based on their importance and impact on the success of the project. This can help ensure that the most critical requirements are addressed first and that the project stays on track. This is especially important when the scope may potentially exceed the timeline. In that case, a prioritized backlog of requirements can facilitate a conversation about taking a phased approach.
- **Document the requirements:** document the app requirements in a clear and concise manner, using a format that is easily accessible to all stakeholders. This can include use cases, user stories, functional specifications, and technical specifications. While there are specialized tools available for managing requirements backlogs and task management, spreadsheets can also work depending on what's available.

Throughout the requirements gathering and design process, it's important to remember one thing: *Design like you know everything but listen like you know nothing.*

5. Cost estimation

In today's digital landscape, mobile applications are a cornerstone of many successful businesses. However, embarking on app development necessitates careful consideration of the associated financial investment. This document serves as a comprehensive guide for estimating the development cost of mobile apps, empowering us to make qualities financial decisions.

Key Determinate of Development Cost:

Several factors influence the overall cost of your mobile app among which we can have the following:

- App Complexity and Functionality: Straightforward apps with basic features will incur lower costs compared to very complicated or detailed apps featuring real-time updates, data synchronization, or complex animations.
- Targeted Platforms: Developing native apps for both iOS and Android platforms typically requires a larger budget than building a cross-platform app that functions on both systems.
- UI/UX Design: Crafting an intuitive and user-centric design necessitates investment in UI/UX expertise. A well-designed interface enhances user engagement and overall app success.
- Backend Development: Apps with extensive data storage and processing needs necessitate a robust backend infrastructure, impacting the cost.
- Development Team Structure: The experience level and geographic location of your development team significantly affect the cost. Hourly rates for developers vary depending on their expertise and region.

Cost Estimation Process

While an exact cost is difficult to point upfront, here's a general process to estimate mobile app development cost:

- Define App Requirements: Clearly define the app's features, functionalities, and target audience.
- Break Down the Development Process: The development process typically involves stages like discovery, design, development, testing, and deployment. Estimate the time required for each stage.
- Consider Development Team Structure: Decide whether to outsource development to a freelancer, agency, or build an in-house team. Factor in their hourly rates.
- Additional Costs: Account for ongoing costs like app store fees, maintenance, and server costs (if applicable).

For example, we can use the following simplified formula to provide a preliminary cost estimate:

Total Development Time (hours) x Hourly Rate = Estimated Cost

For instance, if developing a basic Android app requires 1000 hours at a rate of $40/hour, the estimated cost would be $40,000.

CONCLUSION

In conclusion, mobile app development encompasses different types of apps, including native, web, and hybrid, which utilize programming languages like Swift, Objective-C, Java, Kotlin, HTML, CSS, and JavaScript, along with frameworks such as React Native and Flutter. Architectural patterns like MVC, MVVM, and Clean Architecture help organize the app's structure. Analyzing user requirements is crucial for defining functionalities and designing a seamless user experience. The choice of programming language depends on factors such as the target platform, development resources, performance requirements, and the desired level of access to device capabilities. So overall, it’s just important to remember that work doesn’t stop once your app has been deployed. In fact, it’s only starting.

REFERENCES

1. Amazon from: https://aws.amazon.com/compare/the-difference-between-web-apps-native-apps-and-hybrid-apps/?nc1=h_ls and https://aws.amazon.com/mobile/mobile-application-development/
2. Appivo from https://www.appivo.com/articles/7-key-steps-for-good-app-requirements/
3. *BairesDevBlog*
4. velvetech from https://www.velvetech.com/blog/mobile-app-development-process/
5. https://itcraftapps.com/blog/estimating-mobile-app-development-costs-a-comprehensive-guide/
6. mobidev: https://mobidev.biz/blog/cross-platform-mobile-development-frameworks-comparison
7. Scaler
8. GreekForGreeks : https://www.geeksforgeeks.org/microservices-architecture-on-aws/